MARCIN COPIK <MARCIN.COPIK@INF.ETHZ.CH>

DPHPC: Plotting, Wait-Free

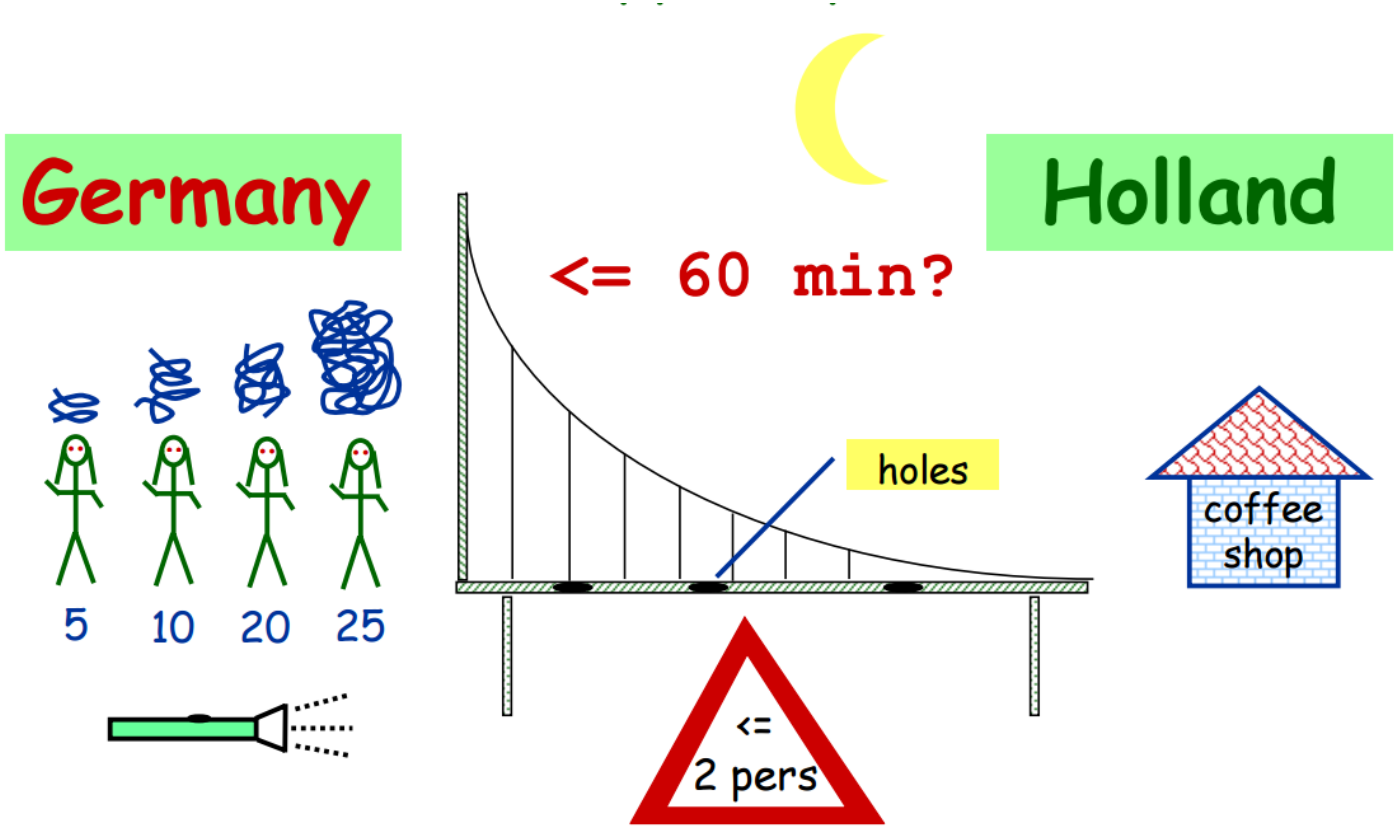
Recitation session

Today's session

1. Previous assignment
2. Lock-free algorithms
3. Good, bad and ugly plots.

Assignment

Hippie problem:
4 Hippies want to cross a bridge. The bridge is fragile, it can only crossed by <= 2 people at a time with a torchlight. The hippies have one torchlight and want to reach the other side within one hour. Due to different degrees of intoxication they require different amounts of time to cross the bridge: 5, 10, 20 and 25 minutes. If a pair crosses the bridge, they can only move at the speed of the slower partner.



Assignment

What do we contain in state?

1. Current time
2. Position of the flashlight
3. Position of each hippie.
4. Remember who's currently travelling.

```
int time;
bit lamp;
bit pos[4];
int bridge[2];
```

Assignment

What do we model?

1. Select two hippies who can travel.

```
int cnt = 0;
        do :: (cnt < 2) -> if
                :: (pos[0] == lamp) →
                        // Hippie 0 is going
                        bridge[cnt] = 0;
                        // Hippie 0 changes his position
                        pos[bridge[cnt]] = !pos[bridge[cnt]];
                        cnt++;
                        printf("hippie %d is going\n", times[0]);
                ...
                fi
        :: (cnt >= 1) -> break;
od;
```

Assignment

What do we model?

2. Determine travel time.

```
// move the lamp in the time the slower one needs
lamp = !lamp;
if
:: ((cnt == 2) && (times[bridge[1]] > times[bridge[0]])) ->
            time = time + times[bridge[1]];
:: else ->
            time = time + times[bridge[0]];
fi
```

Assignment

What do we model?

3. Produce counterexample.

```
assert((time > 60) || ((pos[0] + pos[1] + pos[2] + pos[3]) < 4));
```

Assignment

Live demo.

Problems with locks

- **deadlock**: group of two or more competing processes are mutually blocked because each process waits for another blocked process in the group to proceed
- **livelock**: competing processes are able to detect a potential deadlock but make no observable progress while trying to resolve it
- **starvation**: repeated but unsuccessful attempt of a recently unblocked process to continue its execution

Problems with locks

- **lock-freedom**: at least one algorithm makes progress even if other algorithms run concurrently.
  Implies system-wide progress but not freedom from starvation.

- **wait-freedom**: all algorithms eventually make progress.
  Implies freedom from starvation.

Problems with locks

	Non-blocking (no locks)	Blocking (locks)
Everyone makes progress	Wait-free	Starvation-free
Someone make progress	Lock-free	Deadlock-free

Failure of one thread does not prevent others from working!

Lock-free but wait-free?

```
Object readSomething() {
    return atomicReference.get();
}

void writeSomething(Object new_object) {
    Object old_object;
    do {
        old_object = atomicReference.get();
        // Check if we want to overwrite the latest data (i.e. only write newer or better
data)
        if ( … ) {
            return;
        }
    } while (!atomicReference.compareAndSet(old_object, new_object));
}
```

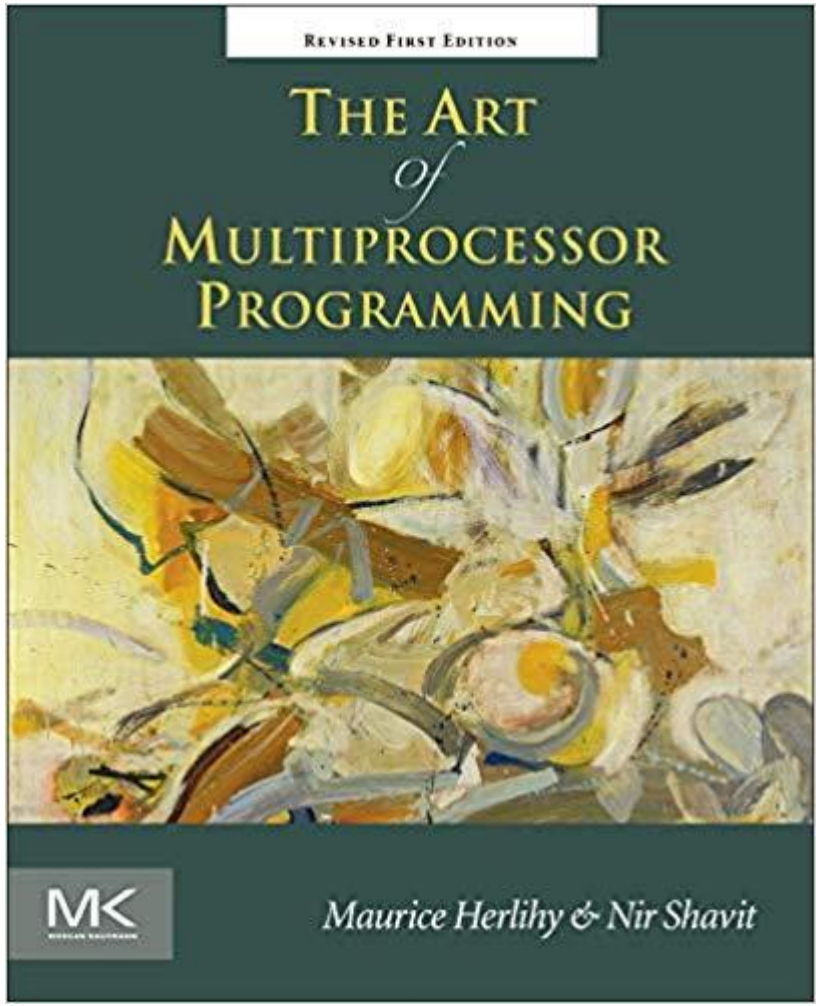
Lock-free update in Java.

```java
@Override
public int nextInt() {
    // get the current seed value
    long next;
    synchronized (this) {
        long orig = state;
        // using recurrence equation to generate next seed
        next = (a * orig + c) & (~0L >>> 16);
        // store the updated seed
        state = next;
    }
    return (int) (next >>> 16);
}
```

```java
@Override
public int nextInt() {
    while (true) {
        // get the current seed value
        long orig = state.get();
        // using recurrence equation to generate next seed
        long next = (a * orig + c) & (~0L >>> 16);
        // store the updated seed
        if (state.compareAndSet(orig, next)) {
            return (int) (next >>> 16);
        }else{
            try {
                Thread.sleep(1);
            } catch (InterruptedException e) {

            }
        }
    }
}
```

More details?



Videos/slides from Parallel Programming class.

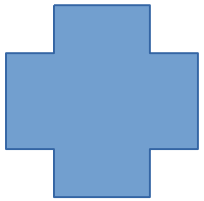
https://spcl.inf.ethz.ch/Teaching/2019-pp/

Benchmarking

What did we learn?

1. Baseline!
2. Report reliable averages with confidence intervals.
3. Average? Median? Percentile?
4. **Present data honestly.**

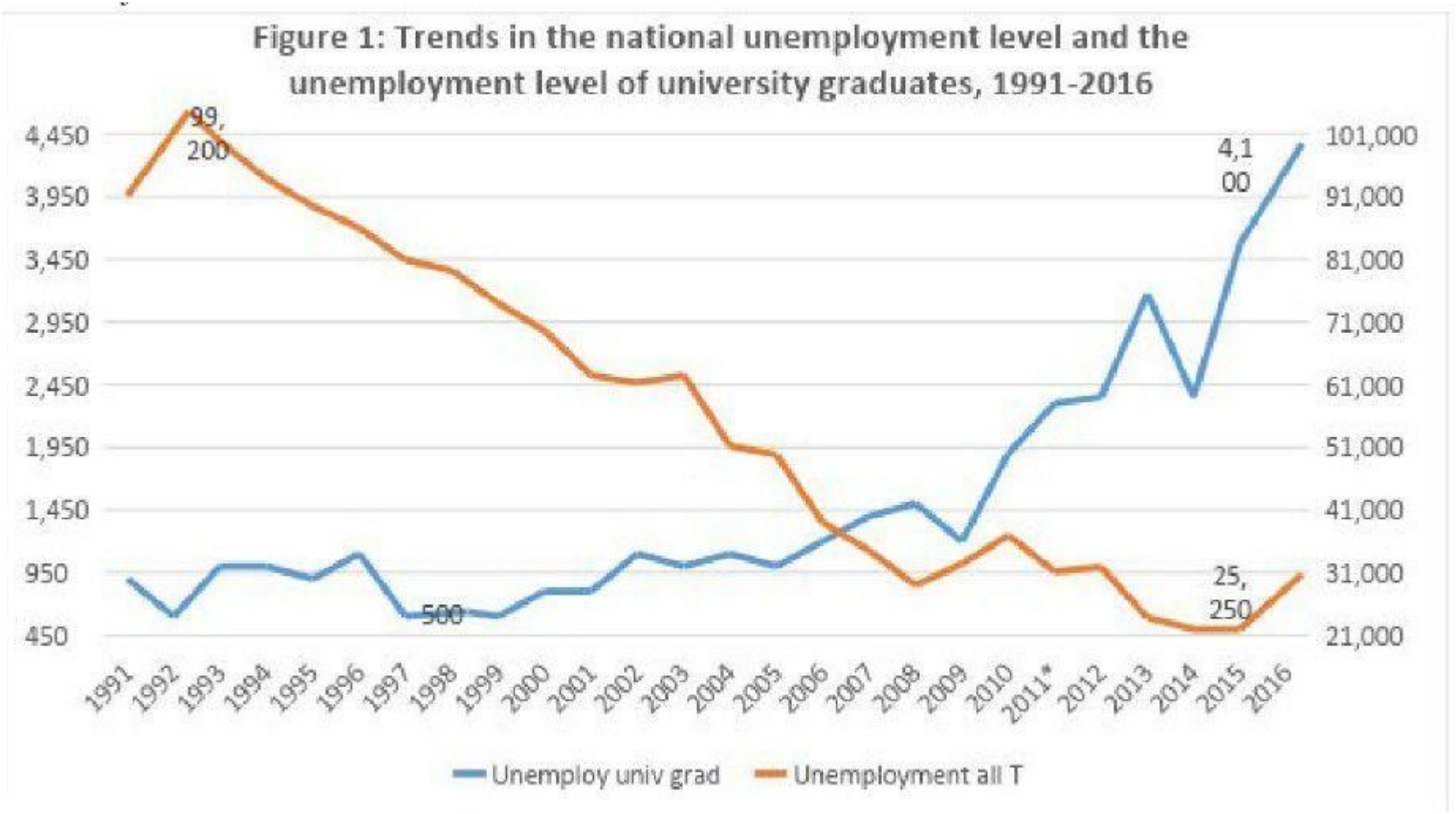
Plotting



Seaborn – nicer and more advanced plots.

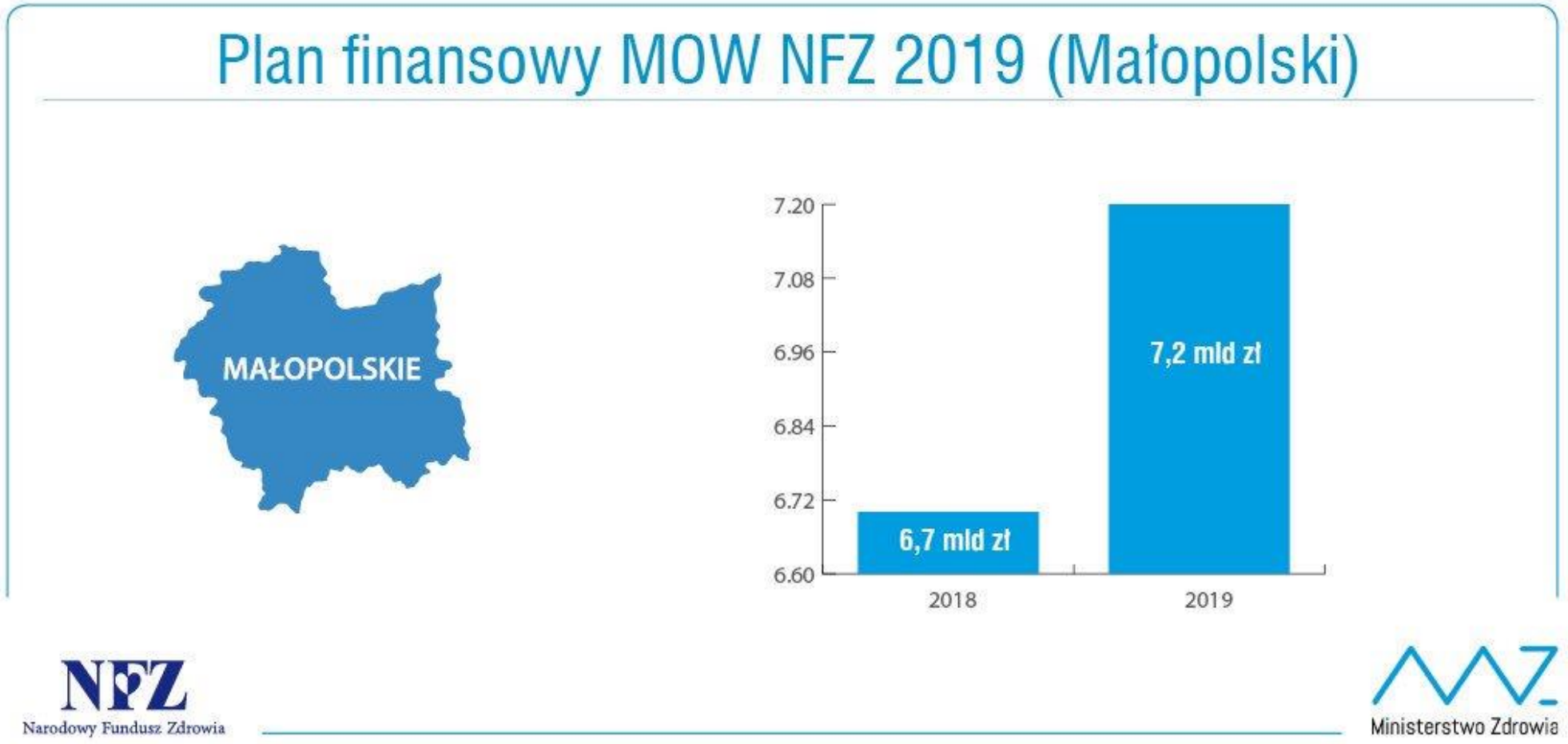
Alternative - R + ggplot2.

Clearly label data



More examples: viz.wtf

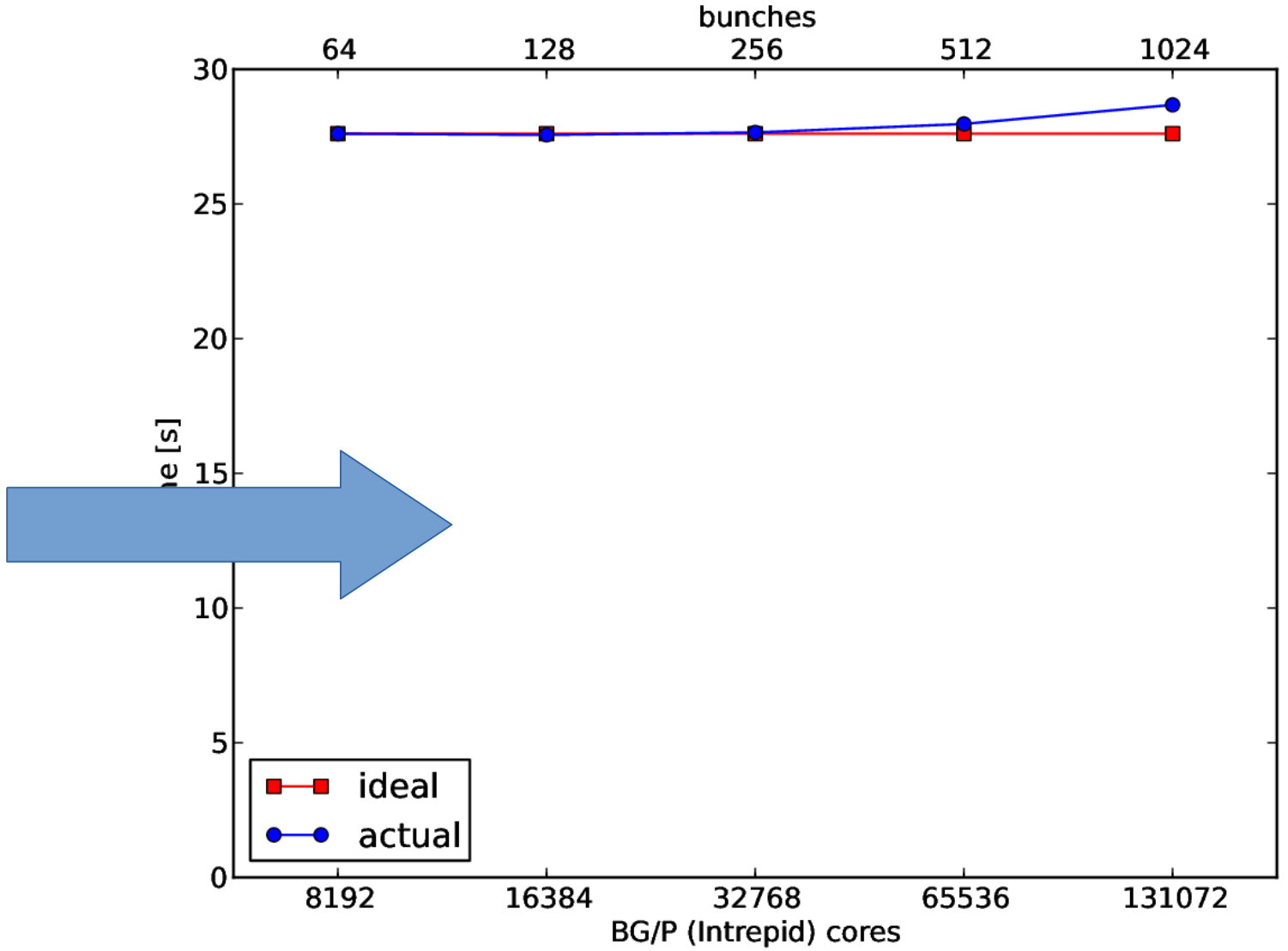
Be honest.



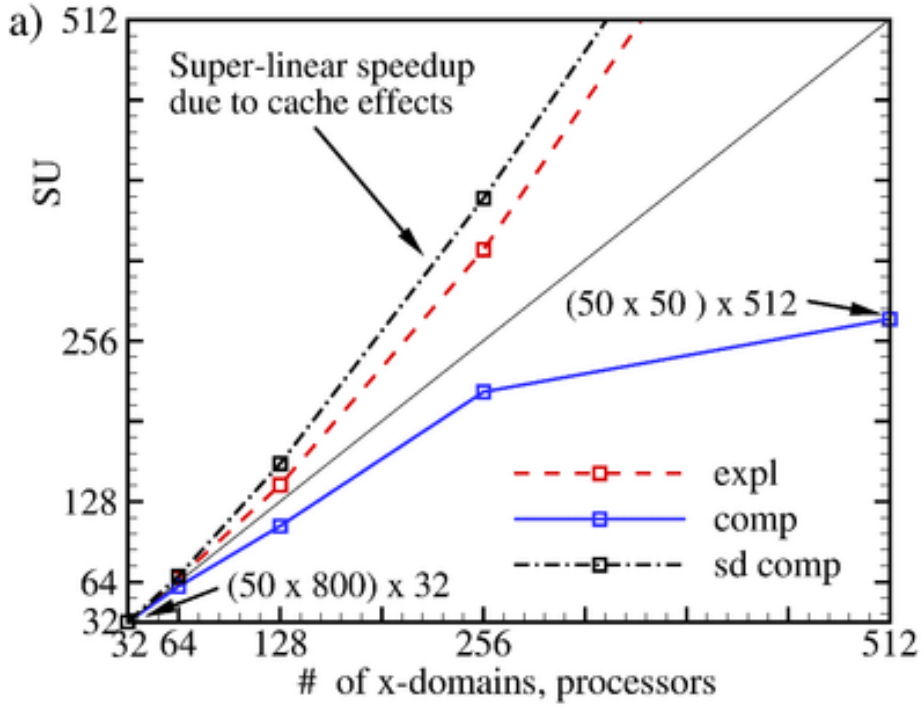
Example: year-to-year spending of national health insurance provider.

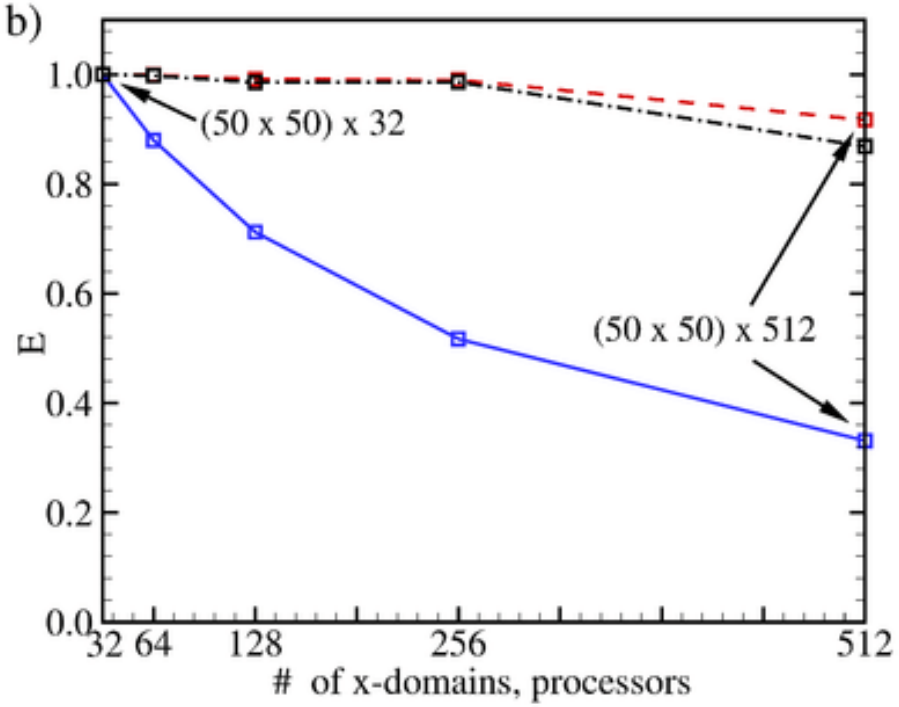
Beautifying plots

What's the point of this space?

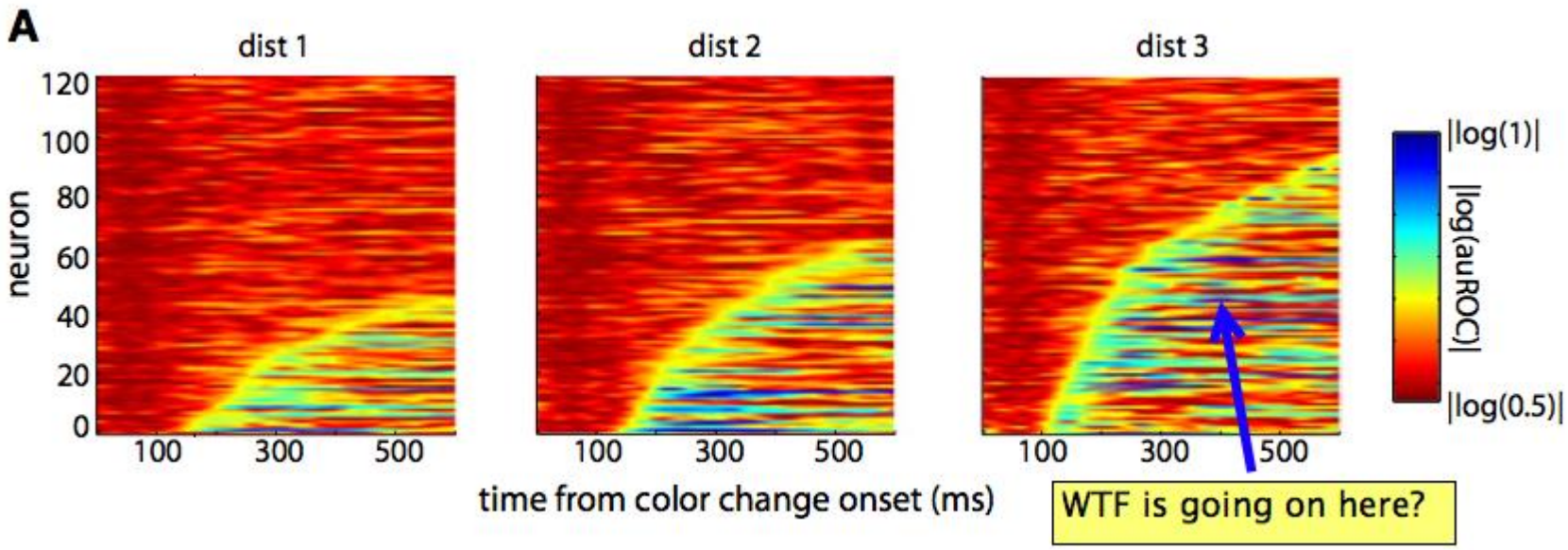


Beautifying plots



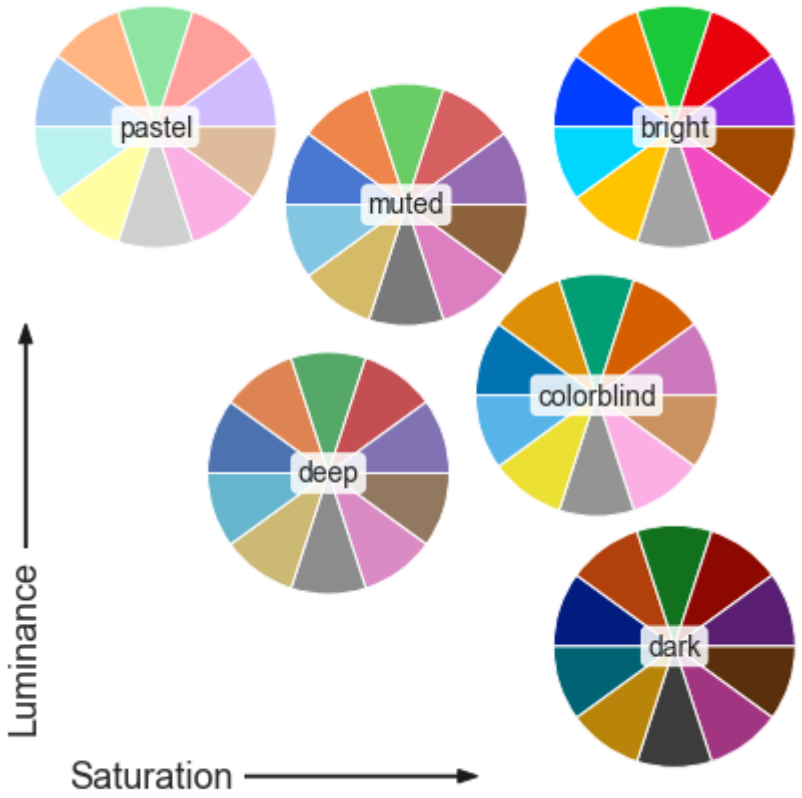


Color Palettes



Source: https://jakevdp.github.io/blog/2014/10/16/how-bad-is-your-colormap/

Color Palettes



How does my plot look after printing (grayscale)?
How does my plot look to a color blind person?

Source: https://seaborn.pydata.org/tutorial/color_palettes.html

Numerical Data

Why? **Some values are hard to plot or hard to read from a plot.**

signal processing concept	algebraic concept (coordinate free)	in coordinates
fi lter	$h \in \mathcal{A}$ (algebra)	$\phi(h) \in \mathbb{C}^{I \times I}$
signal	$s = \sum s_i b_i \in \mathcal{M}$ ($\mathcal{A}$-module)	$\mathbf{s} = (s_i)_{i \in I} \in \mathbb{C}^I$
fi ltering	$h \cdot s$	$\phi(h) \cdot \mathbf{s}$
impulse	base vector $b_i \in \mathcal{M}$	$\mathbf{b}_i = (\dots, 0, 1, 0 \dots)^T \in \mathbb{C}^I$
impulse response of $h \in \mathcal{A}$	$h \cdot b_i \in \mathcal{M}$	$\phi(h) \cdot \mathbf{b}_i = (\dots, h_{-1}, h_0, h_1, \dots)^T \in \mathbb{C}^I$
Fourier transform	$\Delta : \mathcal{M} \to \bigoplus_{\omega \in W} \mathcal{M}_\omega$	$\mathcal{F} : \mathbb{C}^I \to \bigoplus_{\omega \in W} \mathbb{C}^{d_\omega}$ $\Leftrightarrow \phi \to \bigoplus_{\omega \in W} \phi_\omega$
spectrum of signal	$\Delta(s) = (s_\omega)_{\omega \in W} = \omega \mapsto s_\omega$	$\mathcal{F}(\mathbf{s}) = (\mathbf{s}_\omega)_{\omega \in W} = \omega \mapsto \mathbf{s}_\omega$
frequency response of $h \in \mathcal{A}$		$(\phi_\omega(h))_{\omega \in W} = \omega \mapsto \phi_\omega(h)$

Small Guide to Making Nice Tables,
Markus Püschel

signal processing concept	**algebraic concept (coordinate free)**	**in coordinates**
filter	$h \in \mathcal{A}$ (algebra)	$\phi(h) \in \mathbb{C}^{I \times I}$
signal	$s = \sum s_i b_i \in \mathcal{M}$ ($\mathcal{A}$-module)	$\mathbf{s} = (s_i)_{i \in I} \in \mathbb{C}^I$
filtering	$h \cdot s$	$\phi(h) \cdot \mathbf{s}$
impulse	base vector $b_i \in \mathcal{M}$	$\mathbf{b}_i = (\dots, 0, 1, 0, \dots)^T \in \mathbb{C}^I$
impulse response of $h \in \mathcal{A}$	$h \cdot b_i \in \mathcal{M}$	$\phi(h) \cdot \mathbf{b}_i = (\dots, h_{-1}, h_0, h_1, \dots)^T \in \mathbb{C}^I$
Fourier transform	$\Delta : \mathcal{M} \to \bigoplus_{\omega \in W} \mathcal{M}_\omega$	$\mathcal{F} : \mathbb{C}^I \to \bigoplus_{\omega \in W} \mathbb{C}^{d_\omega} \Leftrightarrow \phi \to \bigoplus_{\omega \in W} \phi_\omega$
spectrum of signal	$\Delta(s) = (s_\omega)_{\omega \in W} = \omega \mapsto s_\omega$	$\mathcal{F}(\mathbf{s}) = (\mathbf{s}_\omega)_{\omega \in W} = \omega \mapsto \mathbf{s}_\omega$
frequency response of $h \in \mathcal{A}$	n.a.	$(\phi_\omega(h))_{\omega \in W} = \omega \mapsto \phi_\omega(h)$

Few simple rules

- **Clearly label your data.**
  What's on X axis? What's on Y axis? What's the baseline here?
- **Don't compare different experiments directly.**
- **Clearly communicate changes to representation (log axis, starting point).**
- **Utilize whitespace to include more information.**
  How many bits of data are contained in the image?
- **Don't change data colors between different plots.**
  "Experiment X2 was red and now it's green, I'm lost!"
- **First, make your plot readable and easy to understand. Then make it beautiful.**

Additional resources

Bad plots

- reddit.com/r/dataisugly
- viz.wtf
- junkcharts.typepad.com

Plotting

- https://seaborn.pydata.org/
- https://matplotlib.org/3.1.1/gallery/index.html